Decentralized Optimal Dispatch of Distributed Energy Resources

Alejandro D. Domínguez-García, Stanton T. Cady, and Christoforos N. Hadjicostis

Abstract— **In this paper, we address the problem of optimally dispatching a set of distributed energy resources (DERs) without relying on a centralized decision maker. We consider a scenario where each DER can provide a certain resource (e.g., active or reactive power) at some cost (namely, quadratic in the amount of resource), with the additional constraint that the amount of resource that each DER provides is upper and lower bounded by its capacity limits. We propose a low-complexity iterative algorithm for DER optimal dispatch that relies, at each iteration, on simple computations using local information acquired through exchange of information with neighboring DERs. We show convergence of the proposed algorithm to the (unique) optimal solution of the DER dispatch problem. We also describe a wireless testbed we developed for testing the performance of the algorithms.**

I. Introduction

In a push to increase efficiency and reliability, the structure and functionality of our electricity infrastructure is undergoing radical transformations. Such transformations are enabled by the integration of advanced communication and control; and the integration of distributed energy resources (DERs), e.g., photovoltaic systems, micro-turbines, and plug-in hybrid vehicles, that can provide functionality to the electric grid they are connected to beyond their intended purpose. Among others, these include provision of reactive power for voltage control, and active power for frequency regulation [1].

To enable this added functionality, it is necessary to develop appropriate control and coordination mechanisms. One mechanism relies on a centralized control architecture in which each DER is directly coordinated by and communicates with a central decision maker. An alternative approach is to distribute the decision making, eliminating the need for a central decision maker, but relying on a leader (or other mechanism) that knows the amount of resource that should be collectively provided to achieve the desired functionality. In both cases, the decision making involves solving an *optimal resource allocation* problem for coordinating the DERs to collectively provide a certain amount of a resource (e.g., active or reactive power), subject to their capacity constraints, while minimizing some cost. In this paper, we address the optimal resource allocation problem for DER coordination, and provide distributed algorithms to solve it.

In order to address the distributed optimal dispatch problem, we consider a network of DERs (which we also refer to as nodes), each of which can provide a certain amount of active or reactive power, bounded by upper and lower capacity limits at some cost (namely, quadratic with the amount of resource). By relying on a distributed computation over the network, the objective is for the set of DERs to collectively provide a predetermined amount of the resource, while minimizing the total cost and ensuring operation within individual DER capacity limits. We adopt a very general model for the communication modality between nodes, allowing the exchange of information to be *asymmetric*; specifically, if node i transmits information to another node j, it is not required that node j transmits information to i.

Our starting point to achieve the above objectives is an algorithm that relies on two linear iterations [2], [3]. In this double-iteration algorithm, which we refer to as the *ratio consenus* algorithm, each node j in the network maintains two values y_j and z_j, which we refer to as internal states, and updates them (independently of each other) to be, respectively, a linear combination of node j's own previous internal states, and the previous internal states of its neighboring nodes. It is easy to see that, apart for the initialization of both iterations, the ratio consenus algorithm is a particular case of a gossip-based algorithm proposed in [4], where the weights used by each node are allowed to vary as time evolves. However, the purpose of the algorithm in [4] is for the nodes to compute the average of their initial values.

A good overview on distributed algorithms for optimization problems is provided in [5]; we next discuss more recent works that are specifically related to ours. The authors of [6] propose a linear-iterative algorithm for optimal resource allocation over a network, where the nodes need to collectively provide a fixed amount of a resource, while minimizing their individual costs; the main differences from our approach is that [6] assumes that: the individual component contributions are not upper- and lower-bounded; and communication links are symmetric. In [7], the proposed approach is similar to the one in [6], except for: the equality constraint is substituted by a set of convex inequalities corresponding to individual node constraints, and the communication graph is time-varying. Compared to our work presented herein, [7] imposes no equality constraint on the sum of the individual contributions. Finally, the cost function in [8] is similar to the one in [6], [7], but without equality or inequality constraints.

A. D. Domínguez-García and S. T. Cady are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: {aledan,scady2}@ILLINOIS.EDU.

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus, and also with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: chadjic@UCY.AC.CY.

Domínguez-García and Cady were supported in part by the Consortium for Electric Reliability Technology Solutions and by NSF under grant ECCS-1135598. Hadjicostis was supported in part by the European Commission under grants INFSO-ICT-223844 and PIRG02-GA-2007-224877.

II. Preliminaries

In this section, we formulate the DER optimal dispatch problem, and provide a model that describes the communication network among DERs. We then formulate a linear iterative distributed algorithm defined over the communication network which connects the DERs to solve the optimal dispatch problem in a distributed fashion. The DERs can use the algorithm to, for example, obtain a feasible solution or declare infeasibility of the DER optimal dispatch problem.

A. The Optimal DER Dispatch Problem

Without loss of generality, denote by x_j the amount of resource provided by DER j, where the resource provided could be either active or reactive power with the requirement that all DERs provide the same type of resource, i.e., all the x_j's are either active or reactive power. Let $0 < \underline{x}_j < \overline{x}_j$, for $j = 1, 2, \dots, n$, denote the minimum ($\underline{x}_j$) and maximum ($\overline{x}_j$) capacity limits on the amount of resource x_j that node j can provide. Denote by X the total amount of resource that the DERs must collectively provide, i.e., $X = \sum_{j=1}^{n} x_j$. If we assume that the cost associated to each DER is quadratic, then the optimal DER dispatch problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} \frac{(x_j - \alpha_j)^2}{2\beta_j} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_j = X \\ & 0 < \underline{x}_j \le x_j \le \overline{x}_j, \ \forall j, \end{aligned} \tag{1}$$

where $\alpha_j \le 0$, and $\beta_j > 0$ are real numbers. [When the resource to be provided is active power, the problem in (1) is commonly referred to as economic dispatch (see, e.g., [9]).]

The objective is to solve the optimization program in (1) without relying on a centralized decision maker that has access to all the information defining the problem and posseses adequate computational resources to solve it. Instead we aim to distribute the computation necessary to solve the problem. To this end, we assume that each DER is equipped with a processor that can perform simple computations, and can exchange information with neighboring DERs. In particular, the information exchange between nodes (DERs) can be described by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \dots, n\}$ is the vertex set (each vertex—or node—corresponds to a DER), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, where $(j, i) \in \mathcal{E}$ if node j can receive information from node i. Note that $\mathcal{E}$ could be a proper subset of $\mathcal{V} \times \mathcal{V}$, but we require the graph $(\mathcal{V}, \mathcal{E})$ to be strongly connected. For notational convenience, we allow self-loops for all nodes in $\mathcal{G}$, i.e., $(j, j) \in \mathcal{E}$ for all $j \in \mathcal{V}$. All nodes that can possibly transmit information to node j (including itself) are called its in-neighbors, and are represented by the set $\mathcal{N}_j^- = \{i \in \mathcal{V} : (j, i) \in \mathcal{E}\}$. The nodes that have j as an in-neighbor (including itself) are called its out-neighbors and are denoted by $\mathcal{N}_j^+ = \{l \in \mathcal{V} : (l, j) \in \mathcal{E}\}$; the out-degree of node j is $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$.

B. Centralized Solution to the DER Dispatch Problem

The DER optimal dispatch problem in (1) is convex and has a separable structure [10, pp. 502-506]; its solution can be easily found by solving its Lagrange dual (see, e.g., [10], [11]), which is given by

$$\begin{aligned} \text{maximize} \quad & \lambda X + \sum_{j=1}^{n} f_j(\lambda) \\ \text{subject to} \quad & \lambda \ge 0, \end{aligned} \tag{2}$$

where $f_j(\lambda)$, $\forall j$, are

$$f_j(\lambda) = \begin{cases} \frac{(\underline{x}_j - \alpha_j)^2}{2\beta_j} - \lambda \underline{x}_j, & 0 \le \lambda < \lambda_{2j-1}, \\ -\lambda(\alpha_j + \lambda \frac{\beta_j}{2}), & \lambda_{2j-1} \le \lambda \le \lambda_{2j}, \\ \frac{(\overline{x}_j - \alpha_j)^2}{2\beta_j} - \lambda \overline{x}_j, & \lambda_{2j} < \lambda, \end{cases} \tag{3}$$

with $\lambda_{2j-1} = \frac{\underline{x}_j - \alpha_j}{\beta_j} > 0$ and $\lambda_{2j} = \frac{\overline{x}_j - \alpha_j}{\beta_j} > 0$. The Lagrange dual problem is convex and, in this case, its solution also provides the optimal solution to the primal problem. It is easy to see that $f_j(\cdot) \in \mathcal{C}^1$, $\forall j$, i.e., all the f_j's are continuously differentiable; therefore, the cost function in (2) is also continuously differentiable. Thus, if (2) is feasible, the optimal solution λ^* must satisfy

$$X - \sum_{j=1}^{n} g_j(\lambda^*) = 0, \tag{4}$$

where $g_j(\lambda) = -\frac{df_j(\lambda)}{d\lambda}$, i.e.,

$$g_j(\lambda) = \begin{cases} \underline{x}_j, & 0 \le \lambda < \lambda_{2j-1}, \\ \alpha_j + \lambda \beta_j, & \lambda_{2j-1} \le \lambda \le \lambda_{2j}, \\ \overline{x}_j, & \lambda_{2j} < \lambda. \end{cases} \tag{5}$$

Now, obtaining the value λ^* that satisfies (4) is equivalent to finding the point at which the functions $g(\lambda) := \sum_{j=1}^{n} g_j(\lambda)$ and $h(\lambda) := X$ intersect. With respect to this, it is easy to see from (5) that $g_j(\lambda)$ is a monotonically increasing piecewise linear function, thus $g(\lambda)$ is also a monotonically increasing piecewise linear function with, at most, $2n+1$ line segments defined by the following $2n$ points:

$$\mathcal{U} = \{\lambda_1, \lambda_2, \dots, \lambda_{2n-1}, \lambda_{2n}\}.$$

[The λ_i's do not necessarily form an increasing sequence.] Let $\mathcal{U} = \mathcal{U}^+ \cup \mathcal{U}^-$, where $\mathcal{U}^+$ and $\mathcal{U}^-$ contain the λ_i's in $\mathcal{U}$ that satisfy $g(\lambda_i) \ge X$ and $g(\lambda_i) < X$, respectively. Now, following [12], in order to find λ^*, since $g(\lambda)$ is piecewise linear, we need to find λ^+ and λ^- satisfying

$$\begin{aligned} \lambda^+ &= \arg \min_{\lambda_i \in \mathcal{U}^+} |g(\lambda_i) - X|, \\ \lambda^- &= \arg \min_{\lambda_i \in \mathcal{U}^-} |g(\lambda_i) - X|, \end{aligned} \tag{6}$$

and interpolate between these two values to obtain λ^* satisfying $g(\lambda^*) = X$. Thus,

$$\lambda^* = \lambda^+ - \big[g(\lambda^+) - X\big] \frac{\lambda^+ - \lambda^-}{g(\lambda^+) - g(\lambda^-)}, \tag{7}$$

from where the solution to (1) can be obtained as

$$x_j^* = g_j(\lambda^*), \ \forall j. \tag{8}$$

It is important to note that while (8) provides the unique global solution to the DER optimal problem by solving its Lagrange dual, it requires obtaining the sum of the individual $g_j(\lambda^+)$'s $g_j(\lambda^-)$'s, i.e., a priori, it requires a centralized decision maker with knowledge of all the individual $g_j(\lambda)$'s and X. However, we will show that through a distributed linear iterative algorithm, each DER can obtain (by exchanging information with neighboring nodes) the following quantities: $g(\lambda^+)/X = \sum_{j=1}^n g_j(\lambda^+)/X$ and $g(\lambda^-)/X = \sum_{j=1}^n g_j(\lambda^-)/X$ (and the corresponding λ^+ and λ^-) without the need for learning the individual functions $g_j(\cdot)$, $\forall j$, or the values they take for λ^+, or λ^-, or the value of X. Then, we will show that by slightly rearranging the expression in (7), the computation of the ratios above allows each node to solve the DER optimal dispatch problem in a distributed fashion.

C. The Ratio Consensus Algorithm

Our starting point to solve the decentralized optimal DER dispatch problem is a distributed algorithm that we proposed in our earlier work (see, e.g., [2], [3]). This algorithm, which we refer to as ratio consenus, is a primitive for all distributed algorithms subsequently developed; we next provide a brief overview of its operation.

Consider a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ describing the exchange of information between nodes. Each node j in the network maintains two values y_j and z_j, which we refer to as internal states, and updates them (independently of each other) to be, respectively, a linear combination of node j's own previous internal states, and the previous internal states of its in-neighbors. In particular, for all $k \geq 0$, each node j updates its two internal states as follows:

$$y_j[k+1] = \sum_{i \in \mathcal{N}_j^-} \frac{1}{\mathcal{D}_i^+} y_i[k], \tag{9}$$

$$z_j[k+1] = \sum_{i \in \mathcal{N}_j^-} \frac{1}{\mathcal{D}_i^+} z_i[k], \tag{10}$$

where $\mathcal{D}_i^+$ is the out-degree of node $i \in \mathcal{N}_j^-$, and assuming $z_j[k] > 0, \ \forall k$, node j computes

$$\gamma_j[k] = \frac{y_j[k]}{z_j[k]}. \tag{11}$$

Given the choice of weights in (9)–(10), the following lemma (see [3] for a proof) establishes the convergence of $\gamma_j[k]$ to some constant γ equal for every node j.

Lemma 1: Let $y_j[k]$, $\forall j$, be the result of iteration (9) for some $y_j[0]$, $\forall j$, and $z_j[k]$, $\forall j$, be the result of iteration (10) for some $z_j[0]$, $\forall j$. Then, each node j asymptotically obtains

$$\gamma := \lim_{k \to \infty} \gamma_j[k] = \frac{\sum_{l=1}^n y_l[0]}{\sum_{l=1}^n z_l[0]}.$$

Now, by appropriately choosing a function $h_j : \mathbb{R} \to \mathbb{R}$, and using only local information, each node j can compute, in a distributed fashion, some $x_j = h_j(\gamma)$ of interest, where γ involves information of all the nodes in the system that node j cannot obtain directly from its in-neighbors.

III. Distributed Algorithms for DER Dispatch

In this section, we provide distributed algorithms for dispatching DERs in the following scenarios: i) there are constraints on DER capacity but the objective function is identically zero; ii) there are no constraints on DER capacity, and there is some quadratic cost associated to each DER; and iii) there are constraints on DER upper and lower capacity, and there is some quadratic cost associated to each DER.

A. Constrained Fair-Splitting Dispatch Problem

In (1), if we substitute the quadratic cost function by an objective function that takes value zero if the feasible set defined by the equality and the inequality constraints is nonempty, or ∞ if the feasible set is empty, the problem reduces to the so-called *feasibility problem* (see, e.g., [11]). In our earlier work (see, e.g., [2], [3]), we utilized the algorithm in (9)–(10) to solve the feasibility problem—or declare infeasibility; next, we revisit this idea.

Consider a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ describing the exchange of information between DERs. Without loss of generality, we assume that the total amount of resource X in (1) is known to an entity outside $\mathcal{V}$, referred to as *the leader* who is able to communicate with a set of nodes $\mathcal{L}^+ \subseteq \mathcal{V}$, $\mathcal{L}^+ \neq \emptyset$. Note that $m \equiv |\mathcal{L}^+| \geq 1$. Furthermore, i) the initial conditions in (9) are set to $y_j[0] = X/m - \underline{x}_j$ if $j \in \mathcal{L}^+$, and $y_j[0] = -\underline{x}_j$ otherwise, and ii) the initial conditions in (10) are set to $z_j[0] = \overline{x}_j - \underline{x}_j, \forall j$. Then, as long as $\sum_{j=1}^n \underline{x}_j \leq X \leq \sum_{j=1}^n \overline{x}_j$, it immediately follows from Lemma 1 that the j^{th} DER can asymptotically calculate its contribution x_j as

$$x_j = \underline{x}_j + \gamma(\overline{x}_j - \underline{x}_j), \tag{12}$$

where

$$\gamma = \lim_{k \to \infty} \gamma_j[k] = \lim_{k \to \infty} \frac{y_j[k]}{z_j[k]} = \frac{X - \sum_{l=1}^n \underline{x}_l}{\sum_{l=1}^n (\overline{x}_l - \underline{x}_l)}.$$

Note that (12) satisfies $\underline{x}_j \leq x_j \leq \overline{x}_j$, $\forall j$, and also $\sum_{j=1}^n x_j = X$. Additionally, every node j can independently declare infeasibility if $\gamma > 1$ or $\gamma < 0$. The solution in (12) is not an optimal solution to the problem in (1), but it provides a "fair" split of the total amount of resource X proportional to the "excess" capacity of each DER.

B. Unconstrained Optimal Dispatch Problem

We now consider the case when DER capacity constraints $\underline{x}_j \leq x_j \leq \overline{x}_j$, $\forall j$, in (1) are removed. Then, (3) reduces to $f_j(\lambda) = -\lambda(\alpha_j + \lambda \frac{\beta_j}{2})$, and (5) reduces to $g_j(\lambda) = \alpha_j + \lambda\beta_j$. Thus, (4) reduces to $\sum_{j=1}^n \alpha_j + \lambda^* \sum_{j=1}^n \beta_j = X$, and from (8), it follows that $x_j^* = \alpha_j + \frac{X - \sum_{l=1}^n \alpha_l}{\sum_{l=1}^n \beta_l} \beta_j$, $\forall j$.

Consider a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ describing the exchange of information between DERs. Without loss of generality, we assume that that total amount of resource X in (1) is known to an entity outside $\mathcal{V}$, referred to as *the leader*, who is able to communicate with a set of nodes $\mathcal{L}^+ \subseteq \mathcal{V}$, $\mathcal{L}^+ \neq \emptyset$. Note that $m \equiv |\mathcal{L}^+| \geq 1$. From the fair splitting solution to the feasibility problem given in (12), it is easy to see that the nodes need to run the two linear iterations in (9)–(10) with i) the initial conditions in (9) set to $y_j[0] = \frac{X}{m} - \alpha_j$ if $j \in \mathcal{L}^+$, and $y_j[0] = -\alpha_j$ otherwise, and ii) the initial conditions in (10) are set to $z_j[0] = \beta_j, \forall j$. Then, from Lemma 1, it follows that each node j can obtain the optimal solution as:

$$x_j^* = \alpha_j + \gamma\beta_j, \tag{13}$$

where

$$\gamma = \lim_{k\to\infty} \gamma_j[k] = \lim_{k\to\infty} \frac{y_j[k]}{z_j[k]} = \frac{X - \sum_{l=1}^n \alpha_l}{\sum_{l=1}^n \beta_l}.$$

C. Capacity-Constrained Optimal Dispatch Problem

We now address the most general DER optimal dispatch problem as posed in (1), i.e., including the capacity constraints. As before, consider a strongly connected directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ describing the exchange of information between DERs. Without loss of generality, we assume that the total amount of resource X in (1) is known to an entity outside $\mathcal{V}$, referred to as *the leader*, who is able to communicate with a set of nodes $\mathcal{L}^+ \subseteq \mathcal{V}$, $\mathcal{L}^+ \neq \emptyset$. Note that $m \equiv |\mathcal{L}^+| \geq 1$. We assume that each DER j has knowledge of its capacity limits $\underline{x}_j$ and $\overline{x}_j$ as well as its individual cost $\frac{(x_j-\alpha_j)^2}{2\beta_j}$ and it does not share this information with its out-neighbors.

A key idea to develop a linear-iterative distributed algorithm for solving the DER optimal dispatch problem is to rewrite the solution to the Lagrange dual in (7) as follows

$$\lambda^* = \lambda^+ - \left[\frac{g(\lambda^+)}{X} - 1\right] \frac{\lambda^+ - \lambda^-}{\frac{g(\lambda^+)}{X} - \frac{g(\lambda^-)}{X}}, \tag{14}$$

with $g(\lambda^+) = \sum_{j=1}^n g_j(\lambda^+)$, and $g(\lambda^-) = \sum_{j=1}^n g_j(\lambda^-)$. Note that the calculation of (14), which would allow each node to calculate $x_j^* = g_j(\lambda^*)$, involves sums of the $g_j(\lambda^-)$'s and $g_j(\lambda^+)$'s, but as stated above, each $g_j(\cdot)$ is only known to node j and it will not share it with its out-neighbors. However, by inspection of (14) we note that, for each node j to be able to obtain λ^*, it needs to learn the following quantities: λ^-, λ^+, $\frac{g(\lambda^+)}{X} = \frac{\sum_{l=1}^n g_l(\lambda^+)}{X}$, and $\frac{g(\lambda^-)}{X} = \frac{\sum_{l=1}^n g_l(\lambda^-)}{X}$; next we discuss how a simple message-passing protocol together with the ratio consenus algorithm allows each DER to accomplish this task.

First, each node j needs to obtain $\frac{\sum_{l=1}^n g_l(\lambda)}{X}$, $\forall \lambda \in \mathcal{U}$; and then by following (6), it can obtain λ^-, λ^+, and the corresponding $\frac{\sum_{l=1}^n g_l(\lambda^+)}{X}$, and $\frac{\sum_{l=1}^n g_l(\lambda^-)}{X}$. In order to demonstrate the basic idea, we will explain the simplest case when $\mathcal{U} = \{\lambda_i\}$, and then extend the ideas to any number of λ_i's. Without loss of generality, assume that initially, only node i knows λ_i; then, by broadcasting λ_i, the out-neighbors of i can obtain the value of λ_i, and in turn they can broadcast it to their out-neighbors; continuing in this fashion, in a finite number of steps (bounded by the diameter of the network), each node j will learn λ_i. In addition, each node j maintains two internal states $y_j^{(i)}$ and z_j, which are updated using the ratio consenus algorithm. In particular, once all the nodes posses λ_i, each node j computes $g_j(\lambda_i)$ and sets $y_j^{(i)}[0] = g_j(\lambda_i)$. Additionally, each node j sets $z_j[0] = X/m$ if $j \in \mathcal{L}^+$, and $z_j[0] = 0$ otherwise. Then, it follows from Lemma 1 that

$$\lim_{k\to\infty} \frac{y_j^{(i)}[k]}{z_j[k]} = \frac{\sum_{l=1}^n g_l(\lambda_i)}{X}, \ \forall j. \tag{15}$$

It is important to note that although in the ideas described above we implied that all the nodes first learn λ_i through a message-passing protocol, and then run the ratio consenus algorithm, in practice, this is not necessary, i.e., by a simple modification of the algorithm, the message-passing protocol and the ratio consenus algorithm can get initialized at the same time and convergence of (15) is guaranteed. In particular, at $k = 0$, each node j initializes z_j the same way as above; however, at $k = 0$, only node i possess λ_i, thus only node i sets $y_i^{(i)}[0] = g_i(\lambda_i)$, while every other node j, $j \neq i$, sets $y_j^{(i)}[0] = 0$. [In this scenario, if node i does not pass the value of λ_i to its out-neighbors, then, it is clear that $\lim_{k\to\infty} \frac{y_j^{(i)}[k]}{z_j[k]} = \frac{g_i(\lambda_i)}{X}$.] At $k = 1$, each $l \in \mathcal{N}_i^+$, i.e., each node in the out-neighborhood of node i, obtains the value of λ_i and computes $g_l(\lambda_i)$. Then, at this time step, and only at this time step, each node $l \in \mathcal{N}_i^+$ will add $g_l(\lambda_i)$ to its current $y_l^{(i)}[1]$. [In this scenario, if nodes $l \in \mathcal{N}_i^+$ do not pass the value of λ_i to its out-neighbors, then, it is easy to see that $\lim_{k\to\infty} \frac{y_j^{(i)}[k]}{z_j[k]} = \frac{\sum_{l\in\mathcal{N}_i^+} g_l(\lambda_i)}{X}$.]. Thus, after a finite number of steps, k_0, each node j obtains λ_i and, for some $k \leq k_0$, adds the corresponding $g_j(\lambda_i)$ to its $y_j^{(i)}[k]$; it follows that $\sum_{l=l}^n y_l^{(i)}[k] = \sum_{l=l}^n g_l(\lambda_i)$, $\forall k \geq k_0$ and $\lim_{k\to\infty} \frac{y_j^{(i)}[k]}{z_j[k]} = \frac{\sum_{l=1}^n g_l(\lambda_i)}{X}$.

Now we generalize the ideas above by augmenting the number of auxiliary variables that each node needs to keep track of. In particular, each node j maintains and updates $2n + 1$ variables, $y_j^{(1)}, y_j^{(2)}, \cdots y_j^{(2n)}$, and z_j. Additionally, each node j maintains another set of $2n$ variables $\lambda_j^{(1)}, \lambda_j^{(2)}, \cdots, \lambda_j^{(2n)}$. Initially, each node j sets $y_j^{(j)}[0] = g_j(\lambda_j)$, $y_j^{(l)}[0] = 0$, $\forall l \neq j$, and $z_j[0] = X/m$ if $j \in \mathcal{L}^+$, and $z_j[0] = 0$ otherwise. Additionally, each node j sets $\lambda_j^{(j)}[0] = \lambda_j$, $\lambda_j^{(l)}[0] = 0$, $\forall l \neq j$ and $\lambda_j^{(l)}[-1] = 0$, $\forall l$. Then, at each $k \geq 0$, node j updates $y_j^{(i)}[k]$, $\forall i = 1, 2, \cdots, 2n$, and $z_j[k]$ by performing the following operations:

$$\text{Compute: } u_j^{(i)}[k] = \begin{cases} 0, & \lambda_j^{(i)}[k] = \lambda_j^{(i)}[k-1], \\ 1, & \lambda_j^{(i)}[k] \neq \lambda_j^{(i)}[k-1], \end{cases} \quad \forall i,$$

$$y_j^{(i)}[k+1] = \sum_{l\in\mathcal{N}_j^-} \frac{1}{\mathcal{D}_l^+} y_l^{(i)}[k] + u_j^{(i)}[k] g_j(\lambda_j^{(i)}), \forall i,$$

$$z_j[k+1] = \sum_{l\in\mathcal{N}_j^-} \frac{1}{\mathcal{D}_l^+} z_l[k],$$

$$\lambda_j^{(i)}[k+1] = \max\{\lambda_j^{(i)}[k], \max_{l\in\mathcal{N}_j^-}\{\lambda_l^{(i)}[k]\}\}, \forall i,$$

$$\begin{aligned}\text{Broadcast:}\quad & \frac{1}{\mathcal{D}_j^+} y_j^{(i)}[k+1],\ \forall i,\\ & \frac{1}{\mathcal{D}_j^+} z_j[k+1],\\ & \lambda_j^{(i)}[k+1] \text{ only if } u_j^{(i)}[k]=1, \forall i. \quad (16)\end{aligned}$$

We next provide some additional insights on the operation of the algorithm. From (16), it is clear that at instant k_0, $\lambda_j^{(i)} = 0$ if at every $k \le k_0$, none of the in-neighbors of j has passed the value λ_i to node j. [Each λ_i can be uniquely identified, so even if $\lambda_r = \lambda_s$, for some $r \neq s$, node j can distinguish them and thus properly update the corresponding $\lambda_j^{(r)}$ or $\lambda_j^{(s)}$, respectively.] The first time (namely at $k = k_i$) that node j receives λ_i from one of its in-neighbors, then node j sets $\lambda_j^{(i)}[k_i] = \lambda_i$, computes $g_j(\lambda_j^{(i)}[k]) = g_j(\lambda_i)$ and adds it to $y_j^{(i)}[k_i]$; even if node i receives λ_i again at a later step $k \ge k_i$, it will ignore it. From the above description, it is clear that after a finite number of steps k_m, $\sum_{l=1}^n y_l^{(i)}[k_m] = \sum_{l=1}^n g_l(\lambda_i)$; furthermore, $\sum_{l=1}^n y_l^{(i)}[k] = \sum_{l=1}^n g_l(\lambda_i),\ \forall k \ge k_m$. Then, it follows that for each node j, we have

$$\lim_{k\to\infty} \frac{y_j^{(i)}[k]}{z_j[k]} = \frac{\sum_{l=1}^n g_l(\lambda_i)}{X},\ \forall i. \quad (17)$$

The step left now is for each node to obtain λ^+ and λ^-; there are two options, either node j performs this calculation after a large enough k_l, which ensures that all the $y_j^{(i)}[k_l]/z[k_l]$ have, effectively, reached a steady state, or, at every k, in parallel with the execution of (16), each node j executes the following tasks:

$$\begin{aligned}\text{Compute:}\quad \varepsilon_j^{(i)}[k] &= \frac{y_j^{(i)}[k]}{z_j^{(i)}[k]} - 1,\ \forall i = 1, 2, \cdots, 2n,\\ i_k^+ &= \arg \min_{\{i:\ \varepsilon_j^{(i)}[k] \ge 0\}} |\varepsilon_j^{(i)}[k]|,\\ i_k^- &= \arg \min_{\{i:\ \varepsilon_j^{(i)}[k] < 0\}} |\varepsilon_j^{(i)}[k]|,\\ \lambda_j^*[k] &= \lambda_j^{(i_k^+)}[k] - \varepsilon_j^{(i_k^+)}[k] \frac{\lambda_j^{(i_k^+)}[k] - \lambda_j^{(i_k^-)}[k]}{\varepsilon_j^{(i_k^+)}[k] - \varepsilon_j^{(i_k^-)}[k]},\\ x_j^*[k] &= g_j(\lambda_j^*[k]). \quad (18)\end{aligned}$$

From (17) and (18), it is clear that $\lim_{k\to\infty} \varepsilon_j^{(i)}[k] = \lim_{k\to\infty} \left(\frac{y_j^{(i)}[k]}{z_j^{(i)}[k]} - 1\right) = \frac{\sum_{l=1}^n g_l(\lambda_i)}{X} - 1$, from where it follows that $\lim_{k\to\infty} \lambda_j^{(i_k^+)}[k] = \lambda^+$ and $\lim_{k\to\infty} \lambda_j^{(i_k^-)}[k] = \lambda^-$. Then

$$\begin{aligned}\lim_{k\to\infty} \lambda_j^*[k] &= \lambda_j^{(i_k^+)}[k] - \varepsilon_j^{(i_k^+)} \frac{\lambda_j^{(i_k^+)}[k] - \lambda_j^{(i_k^-)}[k]}{\varepsilon_j^{(i_k^+)} - \varepsilon_j^{(i_k^-)}}\\ &= \lambda^+ - \left[\frac{g(\lambda^+)}{X} - 1\right] \frac{\lambda^+ - \lambda^-}{\frac{g(\lambda^+)}{X} - \frac{g(\lambda^-)}{X}} = \lambda^*, \quad (19)\end{aligned}$$

from where it follows that $\lim_{k\to\infty} x_j^*[k] = x_j(\lambda^*)$.

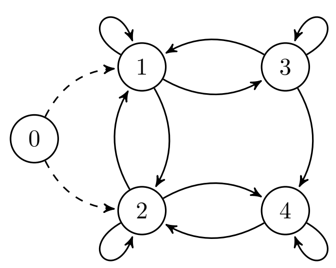

Fig. 1. Graph of 4-node network with leader node.

IV. Experimental Results

In this section, we discuss an implementation of the algorithms presented in Sections III-A and III-B using a hardware testbed consisting of low-complexity devices capable of performing simple computations and communicating wirelessly with other nearby devices. A brief overview of the hardware and software is provided while results are given for two different network configurations.

A. Description of Testbed

Each node in the testbed is centered around an Arduino Mega 2560 microcontroller board. A MaxStream XB24-DMCIT-250 revB XBee module is connected to each Arduino via a SparkFun Electronics XBee shield. The XBee modules provide a wireless link through which the Arduinos exchange packetized information.

A three layered protocol stack is created to facilitate the exchange of information between nodes. The lowest layer of the stack is based upon the ZigBee (IEEE 802.15.4) protocol and is implemented entirely on the XBee modules. The middle layer is a modified version of the XBee-Arduino API and is implemented on the Arduino. The top layer is also implemented on the Arduino and is designed specifically for exchanging information for the distributed algorithms.

In order to ensure convergence of the algorithms when implemented on the testbed, it is imperative that each node begin and end the iterative process at roughly the same time. Thus, upon startup the local clocks of the nodes are synchronized to a common reference, e.g., the clock of the leader node, using the hierarchy referencing time synchronization (HRTS) protocol as proposed in [13]. Furthermore, to ensure that the computation is completed in finite time, the number of iterations performed by the nodes and the period of each iteration is specified *a priori*. During the allotted iteration period, each node broadcasts its state values to the nodes in its out-neighborhood, receives state values from the nodes in its in-neighborhood, and updates its state values according to (9) and (10).

As in all wireless communication, the links between nodes in the testbed are susceptible to interference that can negatively affect communication channel availability. Furthermore, although the XBee protocol seeks to minimize packet collisions, it is possible for nodes to attempt to broadcast their values concurrently, particularly when using a small iteration period. Thus, to ensure convergence despite

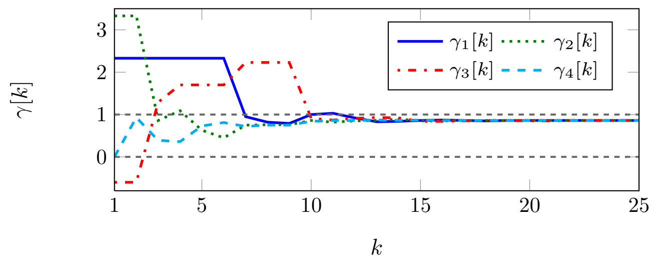

Fig. 2. Evolution of feasible constrained fair-splitting dispatch algorithm.

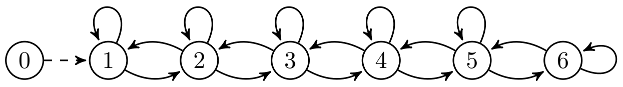

Fig. 3. Graph of 6-node network with leader node.

packet loss, we implement a robustified version of the ratio consensus algorithm proposed in [14].

B. Constrained Fair-Splitting Dispatch Problem

The hardware testbed is used to implement the 4-node network depicted by the graph in Fig. 1. In order to illustrate the ability of individual nodes to determine feasibility, we show results for a case in which the resource demand is within the collective limits of the nodes.

Let the total demand be $X = 1$, and let the minimum and maximum capacities of the nodes be given respectively by $\underline{x} = \left[0.15, 0, 0.15, 0.1\right]^{\mathrm{T}}$ and $\overline{x} = \left[0.3, 0.15, 0.4, 0.25\right]^{\mathrm{T}}$, such that $\sum_{j=1}^{4} \underline{x}_j = 0.4 \leq X \leq \sum_{j=1}^{4} \overline{x}_j = 1.1$. As shown in Fig. 1, the leader node, indexed by 0, can communicate with nodes 1 and 2. Thus, the vectors of initial states are given as $y[0] = \left[0.35, 0.5, -0.15, -0.1\right]^{\mathrm{T}}$, and $z[0] = \left[0.15, 0.15, 0.25, 0.15\right]^{\mathrm{T}}$.

The evolution of $\gamma_j[k]$ for $j = 1, 2, 3, 4$ over 25 iterations is shown in Fig. 2. From the figure, we see that after approximately 15 iterations, all nodes have converged to a solution in which $\gamma = 0.857$. Thus, the nodes determine the solution is feasible and adjust their output according to (12), and we have that $x = \left[0.279, 0.129, 0.364, 0.228\right]^{\mathrm{T}}$.

C. Unconstrained Optimal Dispatch Problem

Consider the 6-node network described by the graph in Fig. 3. Let $X = 1$ and let the vectors of cost coefficients be $\alpha = \left[0.02, 0.1, 0.05, 0.08, 0.12, 0\right]^{\mathrm{T}}$ and $\beta = \left[0.126, 0.108, 0.143, 0.087, 0.109, 0.159\right]^{\mathrm{T}}$. As shown in Fig. 3, the leader, indexed by 0, can only communicate with node 1. Thus the vectors of initial states are given as $y[0] = \left[0.98, -0.1, -0.05, -0.08, -0.12, 0\right]^{\mathrm{T}}$ and $z[0] = \beta$.

Figure 4 shows the evolution of the output of each node, $x_j[k] = \alpha_j + \frac{y_j[k]}{z_j[k]}\beta_j$, over 80 iterations. From the figure, it can be seen that convergence is reached after approximately 60 iterations, enabling the nodes to determine the optimal solution to the dispatch problem. Each node computes its output according to (13) and we have that $x^* = \left[0.128, 0.193, 0.173, 0.155, 0.214, 0.137\right]^{\mathrm{T}}$.

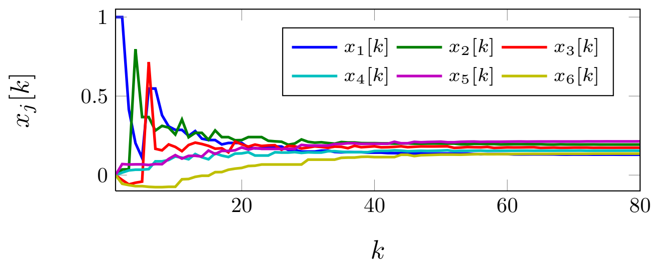

Fig. 4. Evolution of $x_j[k]$, $j = 1, 2, 3, 4, 5, 6$, for the 6-node network.

V. Concluding Remarks and Future Work

In this paper, we addressed the problem of optimally dispatching a set of DERs in a distributed fashion, and showed how the ratio consensus algorithm, which is a linear-iterative algorithm that enables components in a multi-component system to achieve consensus on a certain quantity—namely the ratio of sums of particular values that the nodes possess—is a powerful primitive to solve the problem.

We believe that the algorithms proposed can easily handle scenarios that involve changes in the system during the algorithm execution due to DERs becoming available or unavailable, although we leave the analysis of such scenarios for future work. Future work will also look at extending the algorithms to handle more general convex cost functions; this will likely require a nonlinear interpolation scheme.

References

[1] G. Joos *et al.*, "The potential of distributed generation to provide ancillary services," in *Proc. of IEEE Power Engineering Society Summer Meeting*, vol. 3, Seattle, WA, 2000.

[2] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proc. IEEE SmartGridComm*, 2010, pp. 537–542.

[3] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed algorithms for control of demand response and distributed energy resources," in *Proc. IEEE Conference on Decision and Control*, 2011.

[4] F. Benezit *et al.*, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proc. IEEE International Symposium on Information Theory*, June 2010.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Belmont, MA: Athena Scientific, 1997.

[6] L. Xiao, S. Boyd, and C. P. Tseng, "Optimal scaling of a gradient method for distributed resource allocation," *Journal of Optimization Theory and Applications*, vol. 129, no. 3, pp. 469–488, Jun. 2006.

[7] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, April 2010.

[8] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-Raphson consensus for distributed convex optimization," in *Proc. of IEEE Conference on Decision and Control*, 2011.

[9] A. Bergen and V. Vittal, *Power System Analysis*. Upper Saddle River, NJ: Prentice Hall, 2000.

[10] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 2004.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

[12] M. Madrigal and V. Quintana, "An analytical solution to the economic dispatch problem," *IEEE Power Engineering Review*, vol. 20, no. 9, pp. 52–55, Sep 2000.

[13] H. Dai and R. Han, "Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, pp. 125–139, January 2004.

[14] A. D. Domínguez-García, C. N. Hadjicostis, and N. Vaidya, "Resilient networked control of distributed energy resources," *IEEE Journal on Selected Areas in Comm.*, vol. 30, no. 6, pp. 1137–1148, Jul. 2012.